Event Algebra for Transition Systems Composition - Application to Timed Automata

Elie Fares, Jean-Paul Bodeveix, M Filali

Event Algebra for Transition Systems Composition Application to Timed Automata

Elie FARES
IRIT, Université de Toulouse

Jean-Paul BODEVEIX
IRIT, Université de Toulouse

Mamoun FILALI
IRIT, Université de Toulouse

***Abstract*—Formal specification languages have a lot of notions in common. They all introduce entities usually called processes, offer similar operators, and most importantly define their operational semantics based on labeled transition systems (LTS). However, each language defines specific synchronizing and/or memory structures. For instance, in CSP, the synchronization is defined between identical events, while in CCS and in synchronization vectors-based views it is defined respectively between complementary events or between possibly different events. In this paper, we aim at capturing some similarities of specification languages by defining a label-based composition formal framework. Firstly, we define a high-level synchronization mechanism in the form of an abstract label structure. We then couple this label structure with several compositional operations and properties. Secondly, we introduce an LTS-based behavioral framework and define a unique LTS composition operator which is reused to define syntactic composition of extended transition systems and a compositional semantics.**

I. Introduction

For the past three decades, specification languages such as CSP [17], CCS [16], LOTOS [10], Altarica [5], and BIP [6] have proven valuable in the specification and design of concurrent and distributed systems. The behavioral aspects of these languages share a common base since they all define their operational semantics in terms of labeled transition systems (LTS). Yet, the difference lies in the synchronizing structure of the labels of these systems. For example in CSP the synchronization is defined between two identical events, while in CCS and in synchronization vectors-based views, it is defined respectively between complementary events or between possibly different events. Through the years, the basic versions of some of these languages have been extended by time, memory, and priority notions. Accordingly, other formalisms have emerged in order to model the semantics of these extensions. For example, we can cite Alur and Dill's timed automata [3] and Henzinger et al's timed transition systems [11] that both capture the time addition or the semantic model of [19] used to model the priorities. However, even though the rules of the composition operations of these formalisms are the same in nature (synchronous and asynchronous rules), they are well distinguished in reality, maybe because of the specific attributes that come with each formalism. A distinct composition operation is then introduced for each defined formalism.

In this paper, we aim at capturing some similarities of specification languages by providing a semantic framework for system composition. For this purpose, we introduce a high-level synchronization mechanism in the form of a label structure. The label structure is abstract enough [4] so that both homogeneous and heterogeneous system synchronization could be described. It is equipped with a composition operator which encapsulates the specific composition laws of each language and would further serve as a parameter of the behavioral framework. Thanks to the separation between the composition laws and the behavioral framework, the latter, which is based on LTS, offers a unique LTS composition which is reused to define syntactic composition of timed automata and a compositional semantics. The idea of a label structure or similar constructs is not new since it appears in earlier studies [14], [15], [18], [13]. In [14], [15], the label composition operator appears under a functional form (the same as ours, see Label Structure : Section II) but the authors do not go beyond this definition, while in [18], [13], it appears under a relational form. In these latter studies, the authors are interested in reaching generic semantic rules for process calculi behavioral operators (prefix, choice...). This is orthogonal to our goal that consists in reaching a label-based composition framework for specification languages.

Our contribution can be viewed from two perspectives. One way is to see this work as an abstraction of the composition of different behavioral formalisms via a separation of the labels composition laws of each language and a reuse of the LTS composition. In fact, depending on the language, the label structure is defined and instantiated differently. Using the common framework, one would then proceed by giving the semantics of other behavioral operators of the specification language in question. Another way to see our label structure and their associated operations is as a generalization of the composition functions given and used in [14], [15]. Indeed, we show how such abstract composition functions (or label structures in our terms) may be implemented and instantiated to simulate existing synchronization mechanisms. Most importantly, we push forward this work by giving new definitions, properties, and operations to manipulate such label structures.

The rest of the paper is organized as follows. In the second section, we start by defining the label structure along with its associated properties and operations. In the third section, we define our behavioral framework and show how it reuses the label structure notions. We conclude the paper in the fourth section.

II. Label Structure

We start by describing the labels of transition systems by means of a label structure which is later used as an attribute of transition systems.

Definition 1 (Label Structure): A label structure is a tuple $\langle L, \bowtie \rangle$ where L is a set of labels and ($\bowtie$: $L \times L \nrightarrow L$) is a partial binary composition operator over L.

The function is partial because some composition may be blocked since $\bowtie$ describes exclusively synchronous compositions. The asynchronous aspects are covered later (see LTS composition). Our composition then models the following cases :

1) A successful synchronization between l and l' that results in $l \bowtie l' \in L$.
2) A blocking synchronization between l and l' : $(l, l') \notin \mathbf{dom}(\bowtie)$.

Let the reader not confuse our label structure with other event structuring propositions, namely with the event structures [8]. Event structures model the occurrence of events during the system *execution* via the introduction of a causal dependency relation and a conflict relation between the events. In our case, we introduce a label structure which models the way the labels (i.e., events) are *statically* composed.

Definition 2 (Commutativity of a Label Structure): Given a label structure $LS = \langle L, \bowtie \rangle$, LS is said to be commutative if its composition operator $\bowtie$ is commutative. Formally, for l_1, l_2 and $l_3 \in L$, LS is commutative if :

$$(l_1, l_2) \in \mathbf{dom}(\bowtie) \Rightarrow (l_2, l_1) \in \mathbf{dom}(\bowtie) \wedge l_1 \bowtie l_2 = l_2 \bowtie l_1$$

Definition 3 (Associativity of a Label Structure): Given a label structure $LS = \langle L, \bowtie \rangle$, LS is said to be associative if its composition operator $\bowtie$ is associative. Formally, for l_1, l_2 and $l_3 \in L$, LS is associative if it satisfies the following conditions

1) $(l_1, l_2) \in \mathbf{dom}(\bowtie) \wedge ((l_1 \bowtie l_2), l_3) \in \mathbf{dom}(\bowtie) \Leftrightarrow (l_2, l_3) \in \mathbf{dom}(\bowtie) \wedge (l_1, (l_2 \bowtie l_3)) \in \mathbf{dom}(\bowtie)$. This means that independently of the composition order, they are both defined.
2) $(l_1, l_2) \in \mathbf{dom}(\bowtie) \wedge ((l_1 \bowtie l_2), l_3) \in \mathbf{dom}(\bowtie) \Rightarrow ((l_1 \bowtie l_2) \bowtie l_3) = (l_1 \bowtie (l_2 \bowtie l_3))$. This means that independently of the composition order, they both lead to the same result.

Definition 4 (Stability of a Set of Labels): Given a label structure $LS = \langle L, \bowtie \rangle$ and a label set $G \subseteq L$, we say that G is stable over LS if $\forall (l_1, l_2) \in \mathbf{dom}(\bowtie), l_1 \bowtie l_2 \in G \Rightarrow l_1 \in G \wedge l_2 \in G$, and $\forall (l_1, l_2) \in \mathbf{dom}(\bowtie), l_1 \in G \vee l_2 \in G \Rightarrow l_1 \bowtie l_2 \in G$.

A. Label Structure Examples

a) Time Label Structure: For Δ a time domain, e.g., non negative real numbers, naturals ..., equipped with a binary associative operator $+$ and a neutral element 0, we introduce the time structure TS on the domain Δ. Its composition operator is only defined between identical time labels δ and returns the label itself.

$$TS = \langle \Delta, (\delta_1, \delta_2) \mapsto \delta_1 \textbf{ if } \delta_1 = \delta_2 \rangle$$

b) Basic CSP Synchronizing Structure: Here, we model the case of the completely synchronous composition of CSP. For C a set of communication ports, a synchronizing structure on C is the label structure :

$$Sync_{CSP} = \langle C, (c_1, c_2) \mapsto c_1 \textbf{ if } c_1 = c_2 \rangle$$

The synchronization of two ports of the set C is only defined when these two ports are the same. Otherwise an interleaving occurs.

c) CCS Synchronizing Structure: For C a set of events, $C? = \{c? \mid c \in C\}$ and $C! = \{c! \mid c \in C\}$, this is represented in our label structure as follows :

$$Sync_{CCS} =$$
$$\langle C? \cup C! \cup \{\tau\} \cup \{(c!, c?) \mapsto \tau \mid c \in C\}, \{(c!, c?) \mapsto \tau \mid c \in C\} \rangle$$

B. Label Structure Properties

We give the frequent label structures properties used in this paper.

Property	Definition	Example
Idempotency	$\forall l \in L,$ $(l, l) \in \mathbf{dom}(\bowtie) \wedge l \bowtie l = l$	$Sync_{CSP}, TS$
Unique Composition	$\forall l_1 l_2 \in L, (l_1, l_2) \in \mathbf{dom}(\bowtie) \Rightarrow$ $\forall l \in L, ((l_1 \bowtie l_2), l) \notin \mathbf{dom}(\bowtie)$ $\wedge (l, (l_1 \bowtie l_2)) \notin \mathbf{dom}(\bowtie)$	$Sync_{CCS}$
Diagonality	$\forall l_1 l_2 \in L,$ $(l_1, l_2) \in \mathbf{dom}(\bowtie) \Leftrightarrow l_1 = l_2$	$Sync_{CSP}, TS$

We denote by ***ACI*** the conjunction of the associativity, commutativity and idempotence properties. A label structure fulfilling the ***ACI*** property is seen as a join semi-lattice where $\bowtie$ is interpreted as the join operator and the partial order relation $l \leq l'$ is defined by $l \leq l' \triangleq l \bowtie l' = l'$.

C. Composition of Label Structures

We define the product and the sum of two label structures. The product operation builds new labels as pairs of the composed labels. For example, this is used when composing synchronization and memory access labels. Unlike the product operation, the labels of the sum operation are defined over the union of the composed labels. This is used when composing synchronization and time labels to specify that only one of the events may occur at one time and not simultaneously.

1) Product of Label Structures: Given two label structures $\langle L, \bowtie \rangle$ and $\langle L', \bowtie' \rangle$, their product ranges over the set $P = (L \cup \{\epsilon\}) \times (L' \cup \{\epsilon'\}) \backslash \{(\epsilon, \epsilon')\}$ where ϵ (resp. ϵ') is a new element of L (resp. L') supposed to be neutral [1] for the $\bowtie$ operator of its respective label structure. For l_1, $l_2 \in L$ and l'_1, $l'_2 \in L'$, the composition of $(l_1, l'_1), (l_2, l'_2)$ is defined only if the composition of l_1 and l_2 and the composition l'_1 and l'_2 are both defined.

$$\langle L, \bowtie \rangle \otimes \langle L', \bowtie' \rangle =$$
$$\langle P, \left(\begin{array}{c} (l_1, l'_1), (l_2, l'_2) \mapsto (l_1 \bowtie l_2, l'_1 \bowtie' l'_2) \\ \textbf{if } (l_1, l_2) \in \mathbf{dom}(\bowtie) \wedge (l'_1, l'_2) \in \mathbf{dom}(\bowtie') \end{array} \right) \rangle$$

[1] If L had already a neutral element ϵ, we suppose that $n \bowtie \epsilon = \epsilon \bowtie n = n$.

2) Sum of Label Structures: Given $\langle L, \bowtie \rangle$ and $\langle L', \bowtie' \rangle$, their sum ranges over the union of $L^\bullet \cup {}^\bullet L'$ where $L^\bullet = \{l^\bullet \mid l \in L\}$ and ${}^\bullet L' = \{{}^\bullet l \mid l \in L'\}$.

$$\langle L, \bowtie \rangle \oplus \langle L', \bowtie' \rangle =$$

$$\langle L^\bullet \cup {}^\bullet L', \begin{pmatrix} l_1{}^\bullet, l_2{}^\bullet \mapsto (l_1 \bowtie l_2)^\bullet \textbf{ if } (l_1, l_2) \in \mathbf{dom}(\bowtie) \\ {}^\bullet l_1, {}^\bullet l_2 \mapsto {}^\bullet(l_1 \bowtie' l_2) \textbf{ if } (l_1, l_2) \in \mathbf{dom}(\bowtie') \end{pmatrix} \rangle$$

*Proposition 1 (Preservation of **ACI**):* Given LS and LS', if LS and LS' satisfy one of the ***ACI*** properties then $LS \oplus LS'$and $LS \otimes LS'$ satisfy this same property.

D. Label Structure Transformations

A label structure transformation is used to map labels from a label structure to another. We start by giving the definition of a transformation followed by instances of such transformations.

Definition 5 (Transformation): A transformation f between two label structures $LS_1 = \langle L_1, \bowtie_1 \rangle$ and $LS_2 = \langle L_2, \bowtie_2 \rangle$ is defined as a partial morphism f from LS_1 labels to LS_2 labels such that :

- $\mathbf{ran}(\bowtie_1) \subseteq \mathbf{dom}(f)$.
- $\forall l, l' \in \mathbf{dom}(f), (l, l') \in \mathbf{dom}(\bowtie_1) \Leftrightarrow (f(l), f(l')) \in \mathbf{dom}(\bowtie_2)$.
- $\forall l, l' \in \mathbf{dom}(f), (l, l') \in \mathbf{dom}(\bowtie_1) \Rightarrow f(l \bowtie_1 l') = f(l) \bowtie_2 f(l')$.

We write $f : LS_1 \Rrightarrow LS_2$ to denote such transformations.

1) Basic Transformations: Given two label structures LS_1 and LS_2, we define label structure transformations which are used to embed a label into a sum of labels, destruct a label sum, extend a label to a couple of labels, or also project a couple of labels to an element of the couple. These transformations are given in the following table :

Name	Notation	Signature	Definition
		Embedding	
In_l	${}^{\oplus}\uparrow$	$LS_1 \Rrightarrow LS_1 \oplus LS_2$	$l \mapsto l^\bullet$
In_r	${}^{\oplus}\uparrow$	$LS_2 \Rrightarrow LS_1 \oplus LS_2$	$l \mapsto {}^\bullet l$
		Retraction	
Out_l	${}^{\oplus}\downarrow$	$LS_1 \oplus LS_2 \Rrightarrow LS_1$	$l^\bullet \mapsto l$
Out_r	$\downarrow^{\oplus}$	$LS_1 \oplus LS_2 \Rrightarrow LS_2$	${}^\bullet l \mapsto l$
		Extension	
Ext_l	${}^{\otimes}\uparrow$	$LS_1 \Rrightarrow LS_1 \otimes LS_2$	$l \mapsto (l, \epsilon)$
Ext_r	$\uparrow^{\otimes}$	$LS_2 \Rrightarrow LS_1 \otimes LS_2$	$l \mapsto (\epsilon, l)$
		Projection	
Prj_l	${}^{\otimes}\downarrow$	$LS_1 \otimes LS_2 \Rrightarrow LS_1$	$(l, \epsilon) \mapsto l$
Prj_r	$\downarrow^{\otimes}$	$LS_2 \otimes LS_1 \Rrightarrow LS_2$	$(\epsilon, l) \mapsto l$

It is not difficult to see that the transformation properties (Definition 5) are satisfied by the transformations we have defined. We note that all these transformations are injective.

2) High-Level Transformations: Given the label structures LS, LS_1, LS_2, and $\uparrow_{LS_1}^{LS_2}$ a transformation from LS_1 to LS_2, we define the following four high level transformations :

Name	Signature	tr Transformation
Tr_In_l $(\Uparrow_{\oplus LS}^{\oplus LS})$	$(tr : LS_1 \Rrightarrow LS_2) \rightarrow LS_1 \oplus LS \Rrightarrow LS_2 \oplus LS$	$\begin{cases} l_1{}^\bullet \mapsto tr(l_1)^\bullet \\ {}^\bullet l \mapsto {}^\bullet l \end{cases}$
Tr_In_r $(\Uparrow_{LS\oplus}^{LS\oplus})$	$(tr : LS_1 \Rrightarrow LS_2) \rightarrow LS \oplus LS_1 \Rrightarrow LS \oplus LS_2$	$\begin{cases} {}^\bullet l_1 \mapsto {}^\bullet tr(l_1) \\ l^\bullet \mapsto l^\bullet \end{cases}$
Tr_Ext_l $(\Uparrow_{\otimes LS}^{\otimes LS})$	$(tr : LS_1 \Rrightarrow LS_2) \rightarrow LS_1 \otimes LS \Rrightarrow LS_2 \otimes LS$	$\begin{cases} (l_1, l_2) \mapsto (tr(l_1), l_2) \\ (\epsilon, l_2) \mapsto (\epsilon, l_2) \\ (l_1, \epsilon) \mapsto (tr(l_1), \epsilon) \end{cases}$
Tr_Ext_r $(\Uparrow_{LS\otimes}^{LS\otimes})$	$(tr : LS_1 \Rrightarrow LS_2) \rightarrow LS \otimes LS_1 \Rrightarrow LS \otimes LS_2$	$\begin{cases} (l_1, l_2) \mapsto (l_1, tr(l_2)) \\ (\epsilon, l_2) \mapsto (\epsilon, tr(l_2)) \\ (l_1, \epsilon) \mapsto (l_1, \epsilon) \end{cases}$

We note here that the transformation properties are satisfied by the resulting functions. The previous four transformations also preserve the injectivity of tr.

III. Behavioral Framework

A. Labeled Transition System (LTS)

Definition 6 (Labeled Transition System LTS): Given $LS = \langle L, \bowtie \rangle$, a labeled transition system $\mathcal{L}$ over LS –denoted as $\mathcal{L}_{LS}$– is defined as $\langle Q, Q^0 \subseteq Q, T \subseteq Q \times L \times Q \rangle$ where Q, Q^0, T denote respectively the sets of states, initial states, and transitions. We denote by LTS_{LS} the set of LTSs over LS.

We write $q \xrightarrow{l} q'$ for an element (q, l, q') of T. Furthermore, we define the alphabet of an $\mathcal{L}_{LS}$ –denoted as $\alpha\mathcal{L}_{LS}$– as the set of labels that are actually used by the transitions of $\mathcal{L}_{LS}$: $\alpha\mathcal{L}_{LS} = \{l \in L \mid \exists q\ q',\ q \xrightarrow{l} q' \in T\}$.

Definition 7 (Bisimulation): Given $\mathcal{A}_{LS} = \langle Q_a, Q_a^0, T_a \rangle$ and $\mathcal{C}_{LS} = \langle Q_c, Q_c^0, T_c \rangle$, a relation $R \subseteq Q_c \times Q_a$ defines a simulation between C_{LS} and A_{LS} denoted as $\mathcal{C}_{LS} \precsim_R \mathcal{A}_{LS}$ iff : (1) $\forall q_c^0 \in Q_c^0$, $\exists q_a^0 \in Q_a^0$ such that $(q_c^0, q_a^0) \in R$ and (2) $\forall q_c, q_c', q_a, l$ if $q_c \xrightarrow{l} q_c'$ and $(q_c, q_a) \in R$, $\exists q_a' \in Q_a$ such that $q_a \xrightarrow{l} q_a'$ and $(q_c', q_a') \in R$.

Two LTSs $\mathcal{L}_{LS}$ and $\mathcal{L'}_{LS}$ are said to be bisimilar through the relation $R \subseteq Q \times Q'$ denoted as $\mathcal{L}_{LS} \simeq_R \mathcal{L'}_{LS}$ if $\mathcal{L}_{LS} \precsim_R \mathcal{L'}_{LS}$ and $\mathcal{L'}_{LS} \precsim_{R^{-1}} \mathcal{L}_{LS}$. Furthermore, we say that $\mathcal{L}_{LS}$ and $\mathcal{L'}_{LS}$ are state-bisimilar if transition labels are not required to match.

Definition 8 (LTS Diagonality, Idempotency and Determinism): An LTS is said to be diagonal (resp. idempotent) if the restriction of its label structure to the LTS alphabet is diagonal (resp. idempotent). An LTS is said to be deterministic if whenever $q \xrightarrow{l} q'$ and $q \xrightarrow{l} q''$ then $q' = q''$. In the rest of the paper, this set of LTS properties will be named ***DID***.

1) LTS Composition: Given two LTSs defined over $\langle L, \bowtie \rangle$, a set $S \subseteq L$ denoting the allowed synchronization results, and two sets of labels A_l and A_r denoting respectively the left and right interleaving labels, the label composition function $\bowtie$ is extended to an LTS composition function ${}^{A_l}\langle \bowtie \rangle_S^{A_r}$ as follows:

$$\langle Q_1, Q_1^0, T_1 \rangle {}^{A_l}\langle \bowtie \rangle_S^{A_r} \langle Q_2, Q_2^0, T_2 \rangle = \langle Q_1 \times Q_2, Q_1^0 \times Q_2^0, T \rangle$$

where the set T is defined by the following rules :

$$\frac{q_1 \xrightarrow{l_1}_{T_1} q_1' \quad q_2 \xrightarrow{l_2}_{T_2} q_2' \quad (l_1,l_2) \in \mathbf{dom}(\bowtie) \wedge (l_1 \bowtie l_2) \in S}{(q_1,q_2) \xrightarrow{l_1 \bowtie l_2}_{T} (q_1',q_2')} \text{ SYNC}$$

$$\frac{q_1 \xrightarrow{l_1}_{T_1} q_1' \quad l_1 \in A_l}{(q_1,q_2) \xrightarrow{l_1}_{T} (q_1',q_2)} \text{ INTERLEAVING}_L$$

$$\frac{q_2 \xrightarrow{l_2}_{T_2} q_2' \quad l_2 \in A_r}{(q_1,q_2) \xrightarrow{l_2}_{T} (q_1,q_2')} \text{ INTERLEAVING}_R$$

S is omitted when it is equal to L. In this case, if $A_l = A_r = \emptyset$ then $\langle\bowtie\rangle$ is a fully synchronous composition operator.

Theorem 1 (Bisimulation Compatibility): Given the LTSs $\mathcal{L}_1$, $\mathcal{L}'_1$, $\mathcal{L}_2$, and $\mathcal{L}'_2$ defined over the label structure LS, we have :

$$\mathcal{L}_1 \simeq \mathcal{L}_2 \wedge \mathcal{L}'_1 \simeq \mathcal{L}'_2 \Rightarrow \mathcal{L}_1 \,{}^{A_l}\langle\underset{S}{\bowtie}\rangle^{A_r} \mathcal{L}'_1 \simeq \mathcal{L}_2 \,{}^{A_l}\langle\underset{S}{\bowtie}\rangle^{A_r} \mathcal{L}'_2$$

This theorem allows us to reason by making use of substitution by bisimulation.

Proposition 2 (Synchronous Composition): Given two LTSs $\mathcal{L}_1$ and $\mathcal{L}_2$ defined over the label structure LS, we have $\mathcal{L}_1 \langle\bowtie\rangle \mathcal{L}_2 \simeq \mathcal{L}_1 \langle\underset{S}{\bowtie}\rangle \mathcal{L}_2$ if S is stable over LS, $\alpha\mathcal{L}_1 \subseteq S$, and $\alpha\mathcal{L}_2 \subseteq S$.

Proposition 3 (Commutativity of ${}^{A_l}\langle\underset{S}{\bowtie}\rangle^{A_r}$): Given two LTSs $\mathcal{L}_1$ and $\mathcal{L}_2$ defined over the label structure LS, we have $\mathcal{L}_1 \,{}^{A_l}\langle\underset{S}{\bowtie}\rangle^{A_r} \mathcal{L}_2 \simeq \mathcal{L}_2 \,{}^{A_r}\langle\underset{S}{\bowtie}\rangle^{A_l} \mathcal{L}_1$ if LS is commutative.

Theorem 2 (Associativity of ${}^{A_l}\langle\underset{S}{\bowtie}\rangle^{A_r}$): Given $LS = \langle L, \bowtie \rangle$, the label sets $A_{l_1}, A_{r_1}, A_{l_2}, A_{r_2}, S_1, S_2 \subseteq L$, and the LTSs $\mathcal{L}_{1_{LS}}, \mathcal{L}_{2_{LS}}$, and $\mathcal{L}_{3_{LS}}$. If LS is associative, $S_1, S_2, A_{r_1}, A_{l_2}$ are stable over LS, and either one of the following conditions is satisfied :

1) $A_{r_2} \cap \alpha\mathcal{L}_3 = \emptyset$ and $A_{l_1} \cap \alpha\mathcal{L}_1 = \emptyset$.
2) $A_{l_1} \subseteq A_{l_2}, A_{r_2} \subseteq A_{r_1}, S_2 \cap A_{l_1} = \emptyset$, and $S_1 \cap A_{r_2} = \emptyset$.
3) $A_{l_1} \subseteq A_{l_2}, A_{r_2} \subseteq A_{r_1}, S \subseteq A_{l_1}$, $S \subseteq A_{r_2}$, and $S_1 = S_2$.

We have :

$$\mathcal{L}_1 \,{}^{A_{l_1}}\langle\underset{S_1}{\bowtie}\rangle^{A_{r_1}} (\mathcal{L}_2 \,{}^{A_{l_2}}\langle\underset{S_2}{\bowtie}\rangle^{A_{r_2}} \mathcal{L}_3)$$
$$\simeq$$
$$(\mathcal{L}_1 \,{}^{A_{l_1}}\langle\underset{S_1}{\bowtie}\rangle^{A_{r_1}} \mathcal{L}_2)^{A_{l_2}}\langle\underset{S_2}{\bowtie}\rangle^{A_{r_2}} \mathcal{L}_3$$

The conditions of this theorem are only sufficient conditions and have been selected in order to fit with our needs. Other conditions can be found in other contexts, for example the CSP context [17].

We note how the third set of sufficient conditions satisfy the CCS parallel composition. In this case, this result can be instantiated by taking $S = A_{l_1} = A_{l_2} = A_{r_1} = A_{r_2} = L$ which leads to the following CCS associativity corollary.

Corollary 1 (CCS Associativity): The CCS parallel composition operator is associative.

Other associativity corollaries may also be deduced such as the followings :

Corollary 2: Given an associative label structure $LS = \langle L, \bowtie \rangle$, the label set $S \subseteq L$, the LTSs $\mathcal{L}_{1_{LS}}, \mathcal{L}_{2_{LS}}$, and $\mathcal{L}_{3_{LS}}$ we have :

1) $\mathcal{L}_1 \langle\underset{S}{\bowtie}\rangle (\mathcal{L}_2 \langle\underset{S}{\bowtie}\rangle \mathcal{L}_3) \simeq (\mathcal{L}_1 \langle\underset{S}{\bowtie}\rangle \mathcal{L}_2) \langle\underset{S}{\bowtie}\rangle \mathcal{L}_3$ if $S, S^{\complement}$ are stable over LS [2]. This first proposition is the same as the weak associativity theorem of the CSP generalized parallel operator [17]. The hypothesis added in our context are satisfied by the label structure associated to CSP.
2) $\mathcal{L}_1 \langle\underset{S}{\bowtie}\rangle (\mathcal{L}_2 \langle\bowtie\rangle \mathcal{L}_3) \simeq (\mathcal{L}_1 \langle\underset{S}{\bowtie}\rangle \mathcal{L}_2) \langle\bowtie\rangle \mathcal{L}_3$ if $S, S^{\complement}$ are stable over LS and $\alpha\mathcal{L}_1 \subseteq S$.

Proposition 4 (Idempotency of $\langle\bowtie\rangle$): Given $LS = \langle L, \bowtie \rangle$ and $\mathcal{L}_{LS}$, if $\mathcal{L}_{LS}$ is ***DID*** then $\langle\bowtie\rangle$ is idempotent meaning that $\mathcal{L}_{LS} \langle\bowtie\rangle \mathcal{L}_{LS} \simeq \mathcal{L}_{LS}$.

B. LTS Transformations

The label structure of an LTS may be changed in a composition such as making local a global event (CSP hide) or changing its name (CSP Rename). Here, we consider some LTS labels transformations by extending the label structure transformations to LTS transformations.

Definition 9 (LTS Transformation): Given two label structures LS_1 and LS_2, and a transformation $f : LS_1 \Rrightarrow LS_2$, we define $[f] : \mathcal{L}_{LS_1} \rightarrow \mathcal{L}_{LS_2}$ as :

$$[f]\langle Q, Q_0, \rightarrow_1 \rangle =$$
$$\langle Q, Q_0, \rightarrow_2 = \{(q, f(l), q') \mid l \in \mathbf{dom}(f) \wedge (q, l, q') \in \rightarrow_1\}\rangle$$

Proposition 5 (Transformation Bisimulation Compatibility): Given $LS_1 = \langle L_1, \bowtie_1 \rangle$, $LS_2 = \langle L_2, \bowtie_2 \rangle$, two LTSs $\mathcal{L}_1$ and $\mathcal{L}_2$ both over LS_1, and a transformation $f : LS_1 \Rrightarrow LS_2$, $\mathcal{L}_1 \simeq \mathcal{L}_2 \Rightarrow [f](\mathcal{L}_1) \simeq [f](\mathcal{L}_2)$.

Theorem 3 (Transformation Compositionality): Given $LS_1 = \langle L_1, \bowtie_1 \rangle$, $LS_2 = \langle L_2, \bowtie_2 \rangle$, two LTSs $\mathcal{L}_1$ and $\mathcal{L}_2$ both over LS_1, and an injective transformation $f : LS_1 \Rrightarrow LS_2$ such that $\mathbf{dom}(f)$ is stable over LS_1, we have :

$$[f](\mathcal{L}_1 \,{}^{A_l}\langle\underset{S}{\bowtie}\rangle^{A_r} \mathcal{L}_2) \simeq [f](\mathcal{L}_1)^{f(A_l)}\langle\underset{f(S)}{\bowtie}\rangle^{f(A_r)} [f](\mathcal{L}_2)$$

Proof: We only sketch the proof of the synchronous case. Given LS_1, LS_2, $\mathcal{L}_{LS_1}$, and $\mathcal{L}'_{LS_1}$, we prove that the two sides are bisimilar through the identity relation. The proof is based on showing that each transition of the first system can be found in the second system and vice versa. It is depicted in the following implications which can be read from bottom to top and vice versa from either sides of the parentheses. The main points of this proof are first the use of the stability hypothesis so that we conclude that when $l \in \mathbf{dom}(f)$ then $l_1, l_2 \in \mathbf{dom}(f)$ and conversely, second the use of the injectivity of f in order to connect the two branches of the proof. ■

*Proposition 6 (Preservation of **DID** properties):* Given two label structures LS_1, LS_2, an injective transformation $f : LS_1 \Rrightarrow LS_2$, and an LTS $\mathcal{L}$ over LS_1, each of the

[2] $S^{\complement}$ is the complement of S.

$$\left(\frac{\dfrac{q_1 \xrightarrow{l_1} q_1' \ , \ q_2 \xrightarrow{l_2} q_2' \ , \ l = l_1 \bowtie l_2}{(q_1,q_2) \xrightarrow{l} (q_1',q_2') \ , \ l \in \mathbf{dom}(f)}}{(q_1,q_2) \xrightarrow{f(l)} (q_1',q_2')} \quad \frac{\dfrac{q_1 \xrightarrow{l_1} q_1' \ , \ l_1 \in \mathbf{dom}(f)}{q_1 \xrightarrow{f(l_1)} q_1'} \quad \dfrac{q_2 \xrightarrow{l_2} q_2' \ , \ l_2 \in \mathbf{dom}(f)}{q_2 \xrightarrow{f(l_2)} q_2'}}{(q_1,q_2) \xrightarrow{f(l)=f(l_1)\bowtie f(l_2)} (q_1',q_2')} \ l = l_1 \bowtie l_2 \right)$$

DID properties is preserved by the transformation f. Formally, for P a ***DID*** property we have : $P(\mathcal{L}) \Rightarrow P([f]\mathcal{L})$.

IV. Timed Systems

A. Timed Transition Systems (TTS)

Definition 10 (TTS): Given a label structure $LS = \langle L, \bowtie \rangle$, a Timed Transition System (TTS) over LS is an LTS over $LS \oplus TS$.

Definition 11 (TTS composition): Thanks to the introduction of our label structure, the TTS composition is the composition of the underlying LTSs.

B. Timed Automata (TA)

We consider first a definition of timed automata [3] in which no invariants are associated to its locations (this is close to a timed graph [2] since neither invariants nor committed states are modeled). The transitions are in the form of $guard/event/reset$ where the guards contain a conjunction of constraints represented as clock intervals and the reset actions consist in a set of clocks to be reset. This is represented as a product of three label structures. The first manages the synchronization events, the second manages clock guards, and the third manages the clock reset. In the rest of this paper, we consider a set C of clocks and a time domain Δ (e.g. $\mathbb{R}^+$).

1) Guard Label Structure: Based on the Alur Dill timed automata [3], a guard is a conjunction of interval constraints associated to clocks. Here, this is modeled as a function $C \rightarrow 2^{\Delta}$. The guard label structure is defined as :

$$G \triangleq \langle C \rightarrow 2^{\Delta}, \ (g_1, g_2) \mapsto (c \mapsto g_1(c) \cap g_2(c)) \rangle$$

The guard label structure is ***ACI***.

2) Action Label Structure: Based on the Alur Dill timed automata [3], an action associated to a discrete transition can reset some clocks while keeping the other clocks managed by the current timed automaton unchanged. In order to allow the composition of reset actions, the clocks not managed by a given timed automaton are left undetermined. Consequently an action is modeled by two disjoint sets r denoting the clocks to be reset and u denoting the clocks to be left unchanged. Their composition is defined by respectively the union of the reset sets and the union of the unchanged sets provided that the reset and the unchanged sets are disjoint.

$$\begin{aligned} A \triangleq \langle \ & \{(r,u) \in 2^C \times 2^C \mid r \cap u = \emptyset\}, \\ & ((r_1,u_1),(r_2,u_2)) \mapsto (r_1 \cup r_2, u_1 \cup u_2) \\ & \qquad \textbf{if } r_1 \cap u_2 = r_2 \cap u_1 = \emptyset \rangle \end{aligned}$$

The action label structure is ***ACI***.

3) Timed Automata Label Structure: Given a set of clocks C, we define a timed automaton as an LTS such that its transitions are labeled by communication channels (defined by some label structure LS), guards (the label structure G) and reset actions (the label structure A). For this moment, LS is left undefined and can either model the CCS-based synchronization or the CSP-based one. Given a label structure LS, a timed automaton (TA) over LS is an LTS over $LS \otimes G \otimes A$.

$$TA_{LS} = LTS_{LS \otimes G \otimes A}$$

Definition 12 (TA Composition): Thanks to our label structure, the TA composition is defined as the composition of the underlying LTS systems.

4) TA Semantics: The semantics of a timed automaton is a TTS over $LS \otimes G \otimes A$. In the following we denote GA for $(G \otimes A)$.

Lemma 1: If LS is associative (commutative) [3] then $LS \otimes GA \oplus TS$ is associative (commutative).

Proof: G, A, and TS are associative (commutative). By Proposition 1, $\oplus$ and $\otimes$ preserves the associativity (commutativity). ■

The TA semantics is given via a composition with a clock manager Clk defined over $GA \oplus TS$ (Fig 2).

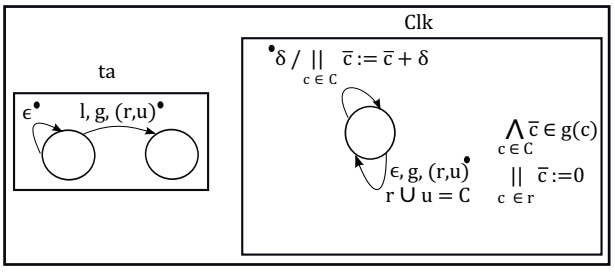

Fig. 1. Semantics of TA via a Composition of Two LTSs

a) Clock Manager: The Clk automaton contains variables (denoted as $\bar{c}$) corresponding to the clocks c of C. It has two types of transitions. The first type correspond to transitions of time evolution labeled by ${}^{\bullet}\delta$ in which after each possible delay all of the clocks are incremented by the amount of this delay. The second type correspond to discrete transitions labeled by $g, (r,u)^{\bullet}$ in which certain clocks are checked against their guard constraints ($\bar{c} \in g(c)$), and clocks belonging to r are reset. In order to impose the determinism of Clk, we suppose that $r \cup u = C$ in Clk, but we synchronize labels l of ta with labels l' of Clk when $l \leq l'$. We recall that

[3]This is verified for both of CCS and CSP.

this partial order relation $\leq$ has been defined from the join operator in Section II-B.

Since Clk is diagonal, idempotent, and deterministic, it follows that $Clk\langle\bowtie\rangle Clk \simeq Clk$ (Proposition 4).

b) Reconstructing the TA Semantics: The TA semantics is defined by means of a composition between the syntactic ta and Clk where ta transmits the clock commands to Clk.

Since the LTS composition is defined over the same label structure, then the label structures on which ta and Clk are defined have to be adapted so that they both become defined over $LS \otimes GA \oplus TS$. More precisely, we use the left embedding transformation for ta and the transformed right extension for Clk. This is formally defined as :

$$[\![ta]\!] = (ta \uparrow_{LS\otimes GA}^{LS\otimes GA\oplus TS} \langle \underset{(LS\otimes GA)^\bullet}{\bowtie} \rangle \ Clk(\Uparrow_{\oplus TS}^{\oplus TS} \uparrow_{GA}^{LS\otimes GA}))$$

Theorem 4 (TA Semantics Compositionality): Given two timed automata ta_1 and ta_2, $[\![ta_1\langle\bowtie\rangle ta_2]\!] \simeq [\![ta_1]\!]\langle\bowtie\rangle[\![ta_2]\!]$.

Proof: This proof is based on proving the bisimulation between the semantics of the composition of ta_1 and ta_2 and the composition of their semantics. We start by unfolding the semantics of the TA composition $[\![ta_1\langle\bowtie\rangle ta_2]\!]$ and by applying a sequence of bisimulations we reach $[\![ta_1]\!]\langle\bowtie\rangle[\![ta_2]\!]$. In the following proof we denote $\langle \underset{(LS\otimes GA)^\bullet}{\bowtie} \rangle$ by $\circledast$ and $\uparrow_{LS\otimes GA}^{LS\otimes GA\oplus TS}$ by $\uparrow_{-}^{-\oplus TS}$.

$$\begin{array}{c}
(ta_1 \uparrow_{-}^{-\oplus TS} \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS})\langle\bowtie\rangle(ta_2 \uparrow_{-}^{-\oplus TS} \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS}) \\
\simeq \{\text{Associativity : Corollary 1.2}\} \\
(ta_1 \uparrow_{-}^{-\oplus TS} \circledast (Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS} \langle\bowtie\rangle (ta_2 \uparrow_{-}^{-\oplus TS} \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS}))) \\
\simeq \{\text{Commutativity : Proposition 3}\} \\
(ta_1 \uparrow_{-}^{-\oplus TS} \circledast ((ta_2 \uparrow_{-}^{-\oplus TS} \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS})\langle\bowtie\rangle Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS})) \\
\simeq \{\text{Associativity : Corollary 1.2}\} \\
(ta_1 \uparrow_{-}^{-\oplus TS} \circledast (ta_2 \uparrow_{-}^{-\oplus TS} \circledast (Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS} \langle\bowtie\rangle Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS}))) \\
\simeq \{\text{Idempotency : Proposition 4}\} \\
(ta_1 \uparrow_{-}^{-\oplus TS} \circledast (ta_2 \uparrow_{-}^{-\oplus TS} \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS})) \\
\simeq \{\text{Associativity : Corollary 1.1}\} \\
((ta_1 \uparrow_{-}^{-\oplus TS} \circledast ta_2 \uparrow_{-}^{-\oplus TS}) \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS}) \\
\simeq \{\text{Synchronous Composition : Proposition 2}\} \\
((ta_1 \uparrow_{-}^{-\oplus TS} \langle\bowtie\rangle ta_2 \uparrow_{-}^{-\oplus TS}) \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS}) \\
\simeq \{\text{Transformation Compositionality : Theorem 3}\} \\
((ta_1\langle\bowtie\rangle ta_2) \uparrow_{-}^{-\oplus TS} \circledast Clk \uparrow_{GA\oplus TS}^{LS\otimes GA\oplus TS})
\end{array}$$

Of course, the hypothesis of the applied results have been verified. Namely, $(LS \otimes GA)^\bullet$ is stable over $LS \otimes GA \oplus TS$, $\alpha ta_1 \uparrow_{LS\otimes GA}^{LS\otimes GA\oplus TS} \subseteq (LS \otimes GA)^\bullet$, ***DID*** is preserved by the transformations, Clk verifies the ***DID*** properties, the product and the transformation are compatible w.r.t bisimulation and Lemma 1. ∎

5) Comparison with Standard TA Semantics: We now state the equivalence between our TA semantics and the standard one. We start by defining the standard TA semantics by the function $[\![\_]\!]_{std} : LTS_{LS\otimes GA} \to LTS_{LS\otimes GA\oplus TS}$ such that $[\![\langle Q, Q^0, \to\rangle]\!]_{std} = \langle Q \times (C \to \mathbb{R}^+), Q^0 \times \{c : C \mapsto 0\}, \to_s\rangle$ where :

$$\frac{q \xrightarrow{(l,(g,(r,u)))} q' \ , \ \bigwedge_{c\in C} v(c)\in g(c) \ , \ \bigwedge_{c\in r} v'(c)=0 \ , \ \bigwedge_{c\in u} v'(c)=v(c)}{(q,v) \xrightarrow{(l,(g,(r,u)))}_s (q',v')}$$

$$\frac{}{(q,v) \xrightarrow{\delta}_s (q, v+\delta)}$$

Theorem 5: Given a timed automaton where each transition is labeled by an action (not by ϵ labels introduced by the label structure product), its standard and proposed (revised) semantics are state-bisimilar through the identity relation.

C. Timed Automata with Invariants

We now add state invariants to timed automata as defined in [12].

1) Invariant Label Structure: Here we consider an invariant to be an upper bound constraint that may be associated to each clock. It is defined as a partial function from clocks to the time domain Δ. The composition of two invariants associates to each clock, when it exists, the minimum of the two bounds. The invariant label structure is defined as :

$$\begin{array}{l}
I \triangleq \langle C \nrightarrow \Delta, \\
\qquad (i_1, i_2) \mapsto (c \mapsto \begin{cases} \min(i_1(c), i_2(c)) \text{ if } c \in \mathbf{dom}(i_1) \cap \mathbf{dom}(i_2) \\ i_1(c) \text{ if } c \in \mathbf{dom}(i_1) \setminus \mathbf{dom}(i_2) \\ i_2(c) \text{ if } c \in \mathbf{dom}(i_2) \setminus \mathbf{dom}(i_1) \end{cases} \rangle\rangle
\end{array}$$

The invariant label structure is ***ACI***.

2) Timed Automata with Invariants Label Structure: Given a set of clocks C, we define a timed safety automaton (TSA) as an LTS such that its transitions are labeled by communication channels (defined by some label structure LS), guards (the label structure G) and reset actions (the label structure A). Furthermore, invariants which are usually attached to locations, are here stored on special looping transitions in order to synchronize with the Invariant Clock controller (IClk in paragraph IV-C3a).

Given a label structure LS, a timed safety automaton over LS is an LTS over $LS \otimes G \otimes A \oplus I$.

$$TSA_{LS} = LTS_{LS\otimes G\otimes A\oplus I}$$

Definition 13 (TSA Composition): Thanks to our label structure, the TSA composition is defined as the composition of the underlying LTS systems.

3) TSA Semantics: The semantics of a timed safety automaton is a LTS over $LS \otimes GA \oplus I \otimes TS$.

Lemma 2: If LS is associative (commutative) then $LS \otimes GA \oplus I \otimes TS$ is associative (commutative).

Proof: G, A, I and TS are associative (commutative). By Proposition 1, $\oplus$ and $\otimes$ preserves the associativity (commutativity). ∎

The TSA semantics is given via a composition with an invariant clock manager $IClk$ defined over $GA \oplus I \otimes TS$ (Fig 2).

a) Invariant Clock Manager : The $IClk$ automaton extends the Clk automaton by constraining the time elapsing. It synchronizes on the invariant specified by the user-provided timed automaton, and on any GA label operator that is equal to the GA label provided by the timed automaton.

Since $IClk$ is diagonal, idempotent, and deterministic, it follows that $IClk\langle\bowtie\rangle IClk \simeq IClk$ (Proposition 4).

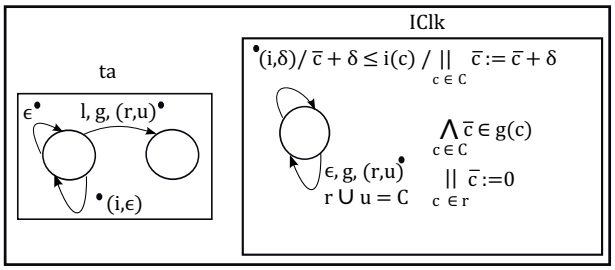

Fig. 2. Semantics of TSA via a Composition of Two LTSs

b) Reconstructing the TSA Semantics: As before, the TSA semantics is defined by means of a composition between the syntactic tsa and $ICl k$. The semantics of a timed safety automaton is reconstructed as follows :

$$[\![tsa]\!] = (tsa(\Uparrow^{LS\otimes GA\oplus}_{LS\otimes GA\oplus} \uparrow^{I\otimes TS}_{I}) \langle\bowtie\rangle \; Clk(\Uparrow^{\oplus I\otimes TS}_{\oplus I\otimes TS} \uparrow^{LS\otimes GA}_{GA}))$$

Theorem 6 (TSA Semantics Compositionality): Given two timed safety automata tsa_1 and tsa_2, $[\![tsa_1 \langle\bowtie\rangle tsa_2]\!] \simeq [\![tsa_1]\!] \langle\bowtie\rangle [\![tsa_2]\!]$.

Proof: This proof is similar to the TA Semantics Compositionality proof. ∎

4) Comparison with Standard TSA Semantics: We now state the equivalence between our TSA semantics and the derived standard one in which a specific encoding of the state invariant is taken into account. We start by defining the standard TSA semantics by the function $[\![\_]\!]_{std} : LTS_{LS\otimes GA\oplus I} \rightarrow LTS_{LS\otimes GA\oplus I\otimes TS}$ such that $[\![\langle Q, Q^0, \rightarrow\rangle]\!]_{std} = \langle Q \times (C \rightarrow \mathbb{R}^+), Q^0 \times \{c : C \mapsto 0\}, \rightarrow_s\rangle$ where :

$$\frac{q \xrightarrow{(l,(g,(r,u)))} q' \;,\; \bigwedge_{c\in C} v(c)\in g(c) \;,\; \bigwedge_{c\in r} v'(c)=0 \;,\; \bigwedge_{c\in u} v'(c)=v(c)}{(q,v) \xrightarrow{(l,(g,(r,u)))}_s (q',v')}$$

$$\frac{q \xrightarrow{i} q \quad \forall c\in C, v(c)+\delta \leq i(c)}{(q,v) \xrightarrow{\delta}_s (q, v+\delta)}$$

Remark that a disjunctive invariant can be modeled using several looping transitions (second rule).

Theorem 7: Given a timed safety automaton where each transition is labeled by an action (not by ϵ labels introduced by the label structure product), its standard and proposed (revised) semantics are state-bisimilar through the identity relation.

V. Towards Giving Life to Label Structures

In this section, we show how the previous semantic definitions could be generalized by attaching behaviors to label structures. For this purpose, we abstract the timed automata semantic construction by associating a behavior to label structures in the form of an LTS. Starting from an LTS built on a given label structure, a syntactic extension can be reached by composing labels with those of a new label structure (for instance, as we have seen, extending LTS to Timed Automata by adding action and guard labels). The corresponding semantics can be defined either by overlaying or by composing the original LTS with the one provided with the extension.

We show that these two methods are equivalent and that, provided some hypothesis, the semantics of the extension is compositional.

A. Semantic Extension

Given a label structure LS_1, we introduce an extension label structure LS_2 supposed to be ***ACI***, and its semantics defined by an LTS $\mathcal{C}$(the controller) over $LS_2 \oplus LS_3$.

We define the extended semantics by the function $[\![\_]\!]^{std}_{\mathcal{C}} : LTS_{LS_1\otimes LS_2} \rightarrow LTS_{LS_1\otimes LS_2\oplus LS_3}$ such that $[\![\mathcal{L} = \langle Q, Q^0, \rightarrow\rangle]\!]^{std}_{\mathcal{C}} = \langle Q \times Q_{\mathcal{C}}, Q^0 \times Q^0_{\mathcal{C}}, \rightarrow_s\rangle$ where :

$$\frac{q \xrightarrow{(l_1,l_2)} q' \;,\; q_{\mathcal{C}} \xrightarrow{l'^{\bullet}_2} q'_{\mathcal{C}} \;,\; l_2 \leq l'_2}{(q,q_{\mathcal{C}}) \xrightarrow{(l_1,l_2)^{\bullet}}_s (q',q'_{\mathcal{C}})}$$

$$\frac{q \xrightarrow{(l,\epsilon)} q'}{(q,q_{\mathcal{C}}) \xrightarrow{(l,\epsilon)^{\bullet}}_s (q',q'_{\mathcal{C}})} \qquad \frac{q_{\mathcal{C}} \xrightarrow{\bullet l} q'_{\mathcal{C}}}{(q,q_{\mathcal{C}}) \xrightarrow{\bullet l}_s (q,q'_{\mathcal{C}})}$$

The semantic LTS is defined over the product of the state space of the syntactic LTS $\mathcal{L}$ and the controller $\mathcal{C}$. Its transitions are built by joining transitions on $LS_1 \otimes LS_2$ and adding transitions over LS_3. In order to allow the composition with user-given LTSs, we add non-determinism through the introduction of the label l'_2 such that $l_2 \leq l'_2$. Coming back to Timed Automata, this corresponds to making at least the resets and the unchanged actions asked by the user. Unreferenced clocks can be freely modified which allows parallel LTSs to impose their own modifications.

This semantics is shown to be equivalent to the following one which reuses the label structure operators

$$[\![\mathcal{L}]\!]_{\mathcal{C}} = (\mathcal{L} \uparrow^{LS_1\otimes LS_2\oplus LS_3}_{LS_1\otimes LS_2} \langle \bowtie_{(LS_1\otimes LS_2)^{\bullet}} \rangle \; \mathcal{C}(\Uparrow^{\oplus LS_3}_{\oplus LS_3} \uparrow^{LS_1\otimes LS_2}_{LS_2}))$$

Theorem 8 (Extension Semantic): Given two LTSs $\mathcal{L}$ over $LS_1 \otimes LS_2$ and $\mathcal{C}$ over $LS_2 \oplus LS_3$ with LS_2 ***ACI***, $[\![\mathcal{L}]\!]^{std}_{\mathcal{C}}$ and $[\![\mathcal{L}]\!]_{\mathcal{C}}$ are state-bisimilar through the identity relation if the following conditions hold :

- transitions of $\mathcal{L}$ are not of the form (l, ϵ),
- the LS_2 labels of the transitions of $\mathcal{C}$ are maximal for $\leq$.

B. Compositionality

The compositionality result concerning the parallel operator of timed automata can be generalized as follows :

Theorem 9 (Generalized Compositionality): Given two LTSs $\mathcal{L}_1$ and $\mathcal{L}_2$ over $LS_1 \otimes LS_2$ and a controller $\mathcal{C}$ over $LS_2 \oplus LS_3$, we have

$$[\![\mathcal{L}_1 \langle\bowtie\rangle \mathcal{L}_2]\!]_{\mathcal{C}} \simeq [\![\mathcal{L}_1]\!]_{\mathcal{C}} \langle\bowtie\rangle [\![\mathcal{L}_2]\!]_{\mathcal{C}}$$

if the following conditions hold :

- LS_1 and LS_3 are associative and commutative.
- LS_2 is ***ACI***.
- $\mathcal{C}$ is ***DID***.

This theorem has a similar proof as the timed automata one. Furthermore, all the hypothesis are satisfied in the timed automata context.

VI. Conclusion

We have presented a formal semantic framework for studying, defining, and manipulating the composition of extended transition systems based on the composition of their labels. The framework is based on the idea of defining a label structure containing a composition operator. Depending on the language in question, a different label structure is defined and thus different composition laws are integrated. The label structure is then used as a parameter of labeled transition systems which describe the common semantic domain of the considered languages. We believe that the suggested parametrization of the behavioral framework is a promising work and may represent, especially with the perspectives we have, the first step towards giving a unified formal semantic framework for different process algebras and specification languages.

In our study, we emphasize on composition operators of process algebras without concentrating on other behavioral operators. In this context, we have pushed forward existing work of similar structures [14], [18], [13] by offering a richer set of operations and properties such as the composition of label structures and transformations between label structures.

Following our technique, the composition of different LTS extensions, whether it is a syntactic model or a semantic model, is captured by a unique composition operation defined on LTS. This is a direct result of the separation between the label structure and the behavioral framework. This result is different than what can be found in the literature since with each system, a different composition operation is provided. This can be seen classically in the composition operations of LTS and TTS. Even though a TTS is exactly an LTS having additionally time transitions, usually its composition operation does not reuse the LTS one.

Furthermore, generic results concerning label structures and LTS transformations are applied to establish well known properties of high-level structures such as the definition of timed automata semantics. We have shown that these semantics match with the standard timed automata semantics and that the timed automata are compositional w.r.t the parallel operator.

We are now working on a dual view of this work which consists in coupling our label structures to states. This will help us to naturally take into consideration state-based mechanisms such as the the committed states of UPPAAL [7]. We are also working on defining the formal semantics of real time languages (BIP [6] and FIACRE [9]). Namely, we are interested in extending our label structure with priorities which are present in all the three cited languages. Another extension is to revisit this work by incorporating some categorical flavor.

Finally, all the theorems related to the presented framework have been validated in the proof assistant Coq. The Coq theory may be found at [1].

Acknowledgement: The authors would like to thank the reviewers for thoroughly reading the paper.

References

[1] http://www.irit.fr/~Jean-Paul.Bodeveix/COQ/LblStr.

[2] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, May 1993.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, Apr. 1994.

[4] F. Arbab. Abstract behavior types: a foundation model for components and their composition. *Sci. Comput. Program.*, 55(1-3):3–52, Mar. 2005.

[5] A. Arnold, G. Point, A. Griffault, and A. Rauzy. The AltaRica formalism for describing concurrent systems. *Fundam. Inf.*, 40(2-3):109–124, 1999.

[6] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in bip. In *SEFM*, pages 3–12. IEEE Computer Society, 2006.

[7] G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal. In M. Bernardo and F. Corradini, editors, *International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–237. Springer Verlag, 2004.

[8] W. Brauer, W. Reisig, and G. Rozenberg, editors. *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, volume 255 of *Lecture Notes in Computer Science*. Springer, 1987.

[9] P. Farail, P. Gaufillet, F. Peres, J.-P. Bodeveix, M. Filali, B. Berthomieu, S. Rodrigo, F. Vernadat, H. Garavel, and F. Lang. FIACRE: an intermediate language for model verification in the TOPCASED environment. In *European Congress on Embedded Real-Time Software*, ERTS'08, 2008.

[10] I. O. for Standardization. *Information processing systems - open systems interconnection - LOTOS - a formal description technique based on the temporal ordering of observational behaviour*. International standard. ISO, 1989.

[11] T. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In J. de Bakker, C. Huizing, W. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 226–251. Springer, 1992. 10.1007/BFb0031995.

[12] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.

[13] T. Hoare and P. O'Hearn. Separation logic semantics for communicating processes. *Electron. Notes Theor. Comput. Sci.*, 212:3–25, Apr. 2008.

[14] H. Hüttel and K. Larsen. The use of static constructs in a model process logic. In A. Meyer and M. Taitslin, editors, *Logic at Botik '89*, volume 363 of *Lecture Notes in Computer Science*, pages 163–180. Springer Berlin Heidelberg, 1989.

[15] K. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In H. Reichel, editor, *Fundamentals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 62–88. Springer, 1995. 10.1007/3-540-60249-6_41.

[16] R. Milner. *Communication and concurrency*. Prentice Hall International, 1995.

[17] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1997.

[18] A. W. Roscoe. On the expressiveness of CSP. Technical report, www.cs.ox.ac.uk/files/1383/complete(3).pdf, 2011.

[19] E. Sekerinski and K. Sere. A theory of prioritizing composition. Technical report, 1996.